Transfer Learning on Inception-Based Deep Learning Models for Civil Infrastructure Defect Classification and Analysis

by

Yash Deshmukh

A thesis presented for the degree of

Master in Science

School of Computing and Mathematics

Keele University

United Kingdom

September 2020

Abstract

The process of identifying defects in civil infrastructures, as the initial phase of the maintenance procedure, is costly and time consuming. The location of these defects varies in the appearance of concrete material, weather conditions, material markings and illumination. Additionally, these defects can overlap with each other. Both of these factors, make defect identification a challenging real-world problem. This dissertation utilises the COncrete DEfect BRidge IMage dataset (CODEBRIM) for multi-class/multi-target classification of five commonly appearing concrete defects. An investigation compares current state of the art (SOTA) Inception Block-based Convolutional Neural Networks (CNN), InceptionV3, InceptionV4, InceptionResNetV2 and Xception, performances, with popular CNNs from the literature and each another, to define a defect classification benchmark for CODEBRIM. The training, validation and testing procedures follow that of the original CODEBRIM paper, where a sigmoid activation function, with a 0.5 threshold has been used with a binary cross entropy loss function. Additionally, the Matthews Correlation Coefficient (MCC) has been used, to measure the quality of binary and multi-class classifications during validation and testing. The experimental results indicate that, all Inception-based CNN architectures substantially outperformed popular CNNs from the literature, with InceptionV3 obtaining the highest test accuracy of 91.0%. The convergence studies indicate that, 200 epochs of training and validation are sufficient for a successful convergence (i.e. As epochs increases, the training and testing accuracies also increase until they become asymptotic) for all CNN models, with the exception of InceptionV4. The contributions of this dissertation involve establishing a defect classification benchmark for CODEBRIM for Inception

Block-based CNNs.

Contents

Chapter 1

Introduction

In Italy, the Genoa bridge has accumulated damage, such as corrosion and cracks, from 1967 to 2018. In 2018, the bridge collapsed. The collapse resulted in 43 human lives lost, property damage and disruptions to commuters [22]. In 2019, 20 people were injured, due to the Nanfang'ao bridge in Taiwan collapsing, which was caused by corrosion of the tension hangers [21]. In 2019, the Hammersmith bridge in London suffered from cracks found in its foundational structures(i.e. footings) [17]. The bridge was declare unfit for regular use and a maintenance procedure is being implemented.

Due to the frequent devastating effects of a bridge collapse, the structural maintaining of bridges has attracted funding from the UK government's Department for Transport (DfT) and interest from many businesses. For example, £1.479 million and £5.6 million in DfT funding has been invested into Albert Road Bridge refurbishment and Blackpool Bridges, respectively [31], whilst businesses such as, Arcadis [3] and PC Richardson & Co Ltd [1], offer bridge maintenance services.

The existing challenge is that, the bridge maintenance process must occur at frequent intervals to reduce the damage accumulation to the lowest possible degree. This process is the existing solution to bridge defect classification and is highly resource-intensive. If these resources are not available, then bridge maintenance is not possible. If this is true, then the bridge will accumulate damages. This increases the probability of bridge collapse occurring. If the process becomes less resource-intensive, then the opposite effect will occur.

The motivation for this dissertation is to use a UAV, to collect bridge defect images (i.e CODEBRIM [34]), and CNNs to effectively and efficiently classify the defects within these images. This is done to reduce the resource-intensiveness of the initial phases of the bridge maintenance process. As a result, bridge maintenance can occur more frequently than in the past, in order to prevent the accumulation of bridge defects and ultimately, reduce the probability of bridge collapse.

The ImageNet dataset consists of 10,000,000 labeled images depicting more than 10,000 object categories [10]. This dataset is used to train image classification neural network models. Test images are presented without any initial annotation, segmentation or labels. These labels will have to produced by the model. Images that are not in the current dataset are collected and labeled for testing the model's performance. The performance is defined as, the models ability to accurately identify the main objects present in the images. This determines the ranking of the model in the ImageNet Large Scale Visual Recognition Challenge (ILSVCR) competition [36]. This competition is used as a benchmark to compare and contrast the image classification abilities of given models.

AlexNet, ZFNet, Inception/ GoogleLeNet, VGG and ResNet are CNNs that have won the ImageNet Large Scale Visual Recognition Challenge (ILSVCR) competition from 2012 to 2015, respectively [27], whilst improving the image classification performance each year. This shows that CNNs are suitable for image classification applications.

Since 2015, CNNs have been improving on their capacity to perform tasks such as, object detection [35], human pose estimation [32], object segmentation [20], video classification [2], object tracking [7] and image classification [19].

This dissertation strives to provide an a definition as to the current SOTA CNN architectures' damage classification performance on a multi-target, multi-class concrete defect image dataset. If this definition does not exist, then the standard for defect classification for civil infrastructures by the current SOTA is not known. As a result, the degree of performance improvement required by the CNN, in order to be used in practice, will not be known. Additionally, the CNN architectures with the highest or weakest performance will not be revealed. This revelation would result in insight into the limitations

of current SOTA CNN architectures. This provides the basis for future research.

Chapter 2

Literature Review

2.1 Convolutional Neural Networks

The Research conducted by AT&T Bell Laboratories Holmdel 1989, demonstrated how constraints, from the class domain, can be integrated into a backpropagation network through the architecture of the network, where the convolution operation and network weights are used [28]. . The architecture is known as, LeNet. The constituents of the LeNet network were 256 input units, three hidden layers and ten output units, where the input of one layer is the output of the previous layer and the outputs at the output layer is comprised of ten units (one per class) and uses place coding. This formed the fundamental CNN architecture. The training procedure was performed within a back-propagation simulator. Additionally, the network weights were updated according to the stochastic gradient process employed. The results were an Top-5 error rate of 0.8% and 0.95% on MNIST database. The research successfully applied backpropagation learning to a large, real world application (i.e handwritten zip code recognition) and achieved SOTA results during its time. This was the major contribution of the research. Since 1989, innovations in processing data input, determining optimum parameters, design strategies, connectivity of the layers of a CNN have resulted in the complexity of CNN architectures and learning methodology have vastly increased [26, 49, 43, 39, 19], where each architecture has improved in its image classification performance.

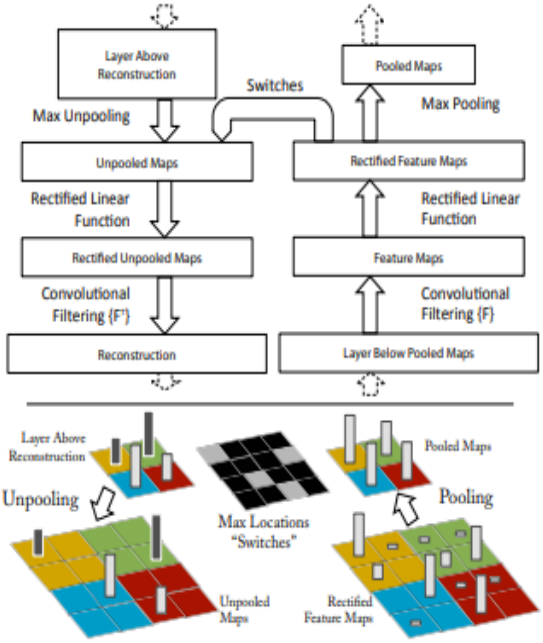


Figure 1: A deconvnet layer (left) attached to a convnet layer (right).

For example, the research conducted by Zeiler et. al used a Deconvolutional Network. This network was used to map the features activities in the hidden layers back to input space (See Figure 1). This is done after the CNN has classified the images. This approach produced a novel visualisation technique that gave insight into the function of intermediate feature layers and the operation of the classifier. The results were an Top-5 error rate of 11.7% on ImageNet database. This is the primary contribution of the paper, which reveals insight into the inner mechanisms of CNNs. These insights were used to make informed decisions about CNN architecture design. This has led to improvements in other applications of CNN. This includes object detection [35], human pose estimation [32], object segmentation [20], video classification [2], object tracking [7] and image classification [19]. Additionally, these improvements have allowed for the effective applications of CNN, when processing large, complex, multi-class, multi-target, heterogeneous datasets [34].

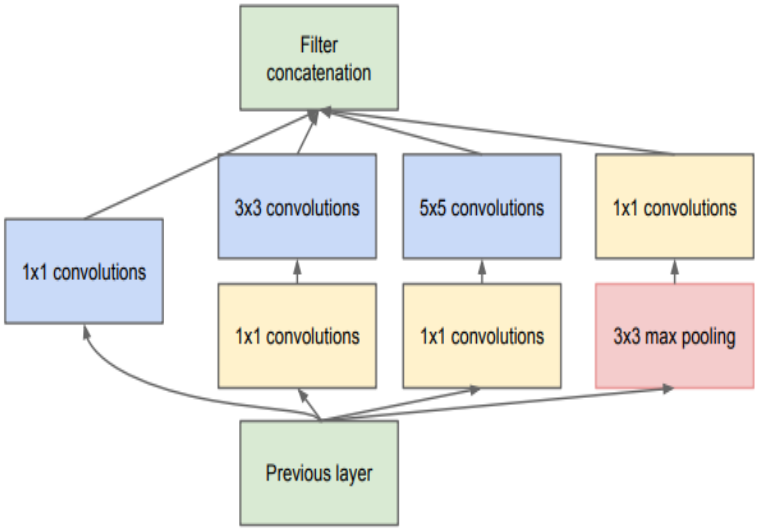


Figure 2: Inception module.

A CNN architecture of particular interest is the Inception/ GoogleLeNet architecture. The architecture for this deep CNN (DCNN) was proposed in 2014 by Szegedy et. al [43]. The architecture, utilises the Hebbian principle [14] and multi-scale processing, in order to reduce the number of dimensions of Inception blocks. This results in higher network depth and width, whist maintaining a constant computational budget. Specifically, the Inception block used. See Figure 2, within the GoogleLeNet architecture, contained the novel use of splitting and merging multiple branches across the network. This allowed for a further refinement of the feature extraction pipeline section and an improved utilisation of computational resources, when compared to CNN architectures of the past. The results were an Top-5 error rate of 6.7% on ImageNet database. This is the primary contribution of this research.

However, the novel usage of splitting and merging multiple branches across a network increases the depth and width of a network. This may result in the Exploding/ Vanishing Gradient problem [5, 15]. This problem is, when the derivatives or slopes become very large or very small, during the training process. If this occurs then training a deep CNN becomes difficult. Research conducted by He et. al proposed a residual learning framework to ease the difficulty associated with training deep CNNs [19].

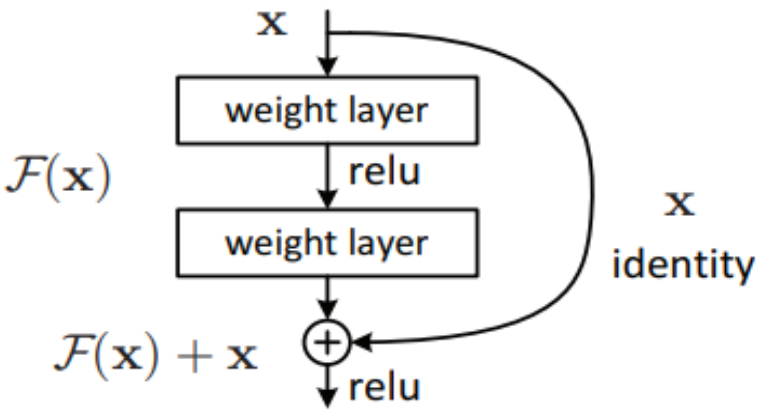


Figure 3: Residual block.

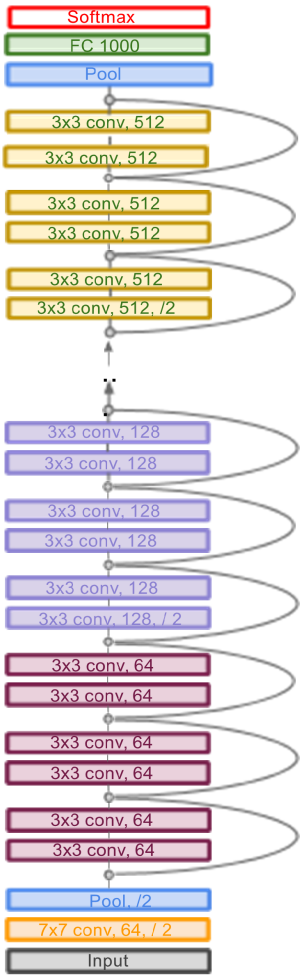


Figure 4: Residual network with unspecified number of parameter layers.

This is done by using Residual Blocks (See Figure 3), where these blocks consist of, the activation from the previous layer being added to the activation of a deeper layer in the same network. See Figure 4. This is expressed as a Skip Connection. These

skip connections result in the exploding/ vanishing gradient problem being significantly reduced, which allows for the effective training of deeper CNNs. The results were an Top-5 error rate of 3.6% and 6.43% on ImageNet and CIFAR-10 databases, respectively. This was the primary contribution of the paper. Due to this, CNN architecture designs, that incorporate residual blocks, suffered less from the vanishing/ gradient problem. This allowed for CNN architectures such as, ResNeXt [47], Xception [9], Pyramidal Net [18] and Inception-ResNet [42] (See Figure 17) to be constructed.

In summary, CNN has improved on its capacity to learn and perform well on a variety of applications, especially image classification. As a result, CNNs have received interest from researchers and are expected to improve, as hardware and new approaches are developed.

CNN Architecture (Year)	Primary Contribution	Unique Methodology (compared to previous SOTA)	Database Used & Top-5 Error Rate
LeNet (1998) [28]	First major CNN architecture	Image Preprocessing,CNN layers, Fully connected layers, Softmax	MNIST: 0.8, 0.95
AlexNet (2012) [26]	Increased depth and uses ReLU activation	Local response normalisation	ImageNet: 16.4
ZFNet (2014) [49]	Hidden layer visualisation	Contrast normalisation	ImageNet: 11.7
VGG (2014) [39]	Increased depth to 16-19 layers, improves CNN performance	CNN blocks	ImageNet: 7.3
Inception/ GoogleLeNet (2015) [43]	Splitting and merging of CNN layers in blocks	Inception Module, Batch Normalisation, Global average pooling	ImageNet: 6.7
Inception-V3 (2015) [44]	Scale networks to utilise the added computation optimally	n/a	ImageNet: 3.5, Single-Crop: 3.58, Mutli-Crop: 5.6
ResNet (2016) [19]	Residual connections, as a solution to exploding/ vanishing problem	Residual connections, Max pooling	ImageNet: 3.6, CIFAR-10: 6.43
ResNeXt (2017) [47]	Repeated blocks results in homogeneous, multi-branch architecture with few hyperparameters	Inception module and Residual connections	CIFAR-10: 3.58, CIFAR-100: 17.31,ImageNet: 4.4
Xception (2017) [9]	Inception modules have been replaced with depthwise separable convolutions	Xception Block	ImageNet: 0.055
PyramidalNet (2017) [18]	Gradually increasing the feature map dimension at all units to involve as many locations as possible	PyramidalNet approach, ReLU and Batch Normalisation in novel positions	CIFAR-10: 4.7, CIFAR-100: 3.48,ImageNet: 17.01
Inception-V4 (2016) [42]	Increasing branches compared to Inception-V3 and asymmetric filters used	Inception-V3 and Residual connections	ImageNet: 4.01
Inception-ResNet (2016) [42]	Inception-V4 with residual connections	n/a	ImageNet: 3.52

Table 1: Research Summary of CNN Architectures, which highlights the major contributions and the corresponding classification results on benchmark datasets.

CNN Architecture	Research Strengths	Research Weaknesses
LeNet	Automatic hierarchical-feature learning	Low level feature extraction only
AlexNet	Deep and wide CNN architecture, Regularisation used, GPU training accelerator	Inefficient use of artificial neurons
ZFNet	Hidden layer visualisation, parameter refinement	Computationally expensive visualisation technique
VGG	receptive field	Computationally expensive deeply connected layers
Inception/ GoogleLeNet	Multi-factor filters, Splitting and merging CNN branches	Cumbersome parameter adjustments, due to heterogeneous architecture
Inception-V3	asymmetric features used lowers computational costs of DCNN	Cumbersome parameter adjustments, due to heterogeneous architecture
ResNet	Residual connections, solution to exploding/ vanishing gradient problem	Added complexity to CNN architecture
ResNeXt	Grouped convolutions, homogeneous architecture design, therefore simple parameter adjustments	Computationally expensive
Xception	Decomposition of filter-based learning, reduces learning difficulty, Depth-wise convolutions	Computationally expensive
PyramidalNet	Increasing the width gradually per unit, therefore accounts for varying dimensionalities	High computational expense, if width continues to increase indefinitely.
Inception-V4	Improved design and performance compared to Inception-V3	Cumbersome parameter adjustments, due to heterogeneous architecture
Inception-ResNet	Inception block used in conjunction with Residual connections	n/a

Table 2: Research Summary of CNN architectures Strengths and Weaknesses, where the weaknesses highlight potential areas of improvements to the CNN architectures.

2.2 Automatic defect detection in civil infrastructures

The automatic detection of defects and bridges will reduce the resource-intensiveness of the initial phases of the bridge maintenance process. As a result, bridge maintenance can occur more frequently than in the past, in order to prevent the accumulation of bridge defects and ultimately, reduce the probability of bridge collapse. For this reason, the automatic detection of bridge defects has become a major research interest over the years [37, 48, 46, 34]. Additionally, CNN performance on image classification and segmentation has improved over the years [43, 42]. As such, CNNs are being applied to the automatic detection of defects in civil infrastructures [12].

Research conducted by Shi et. all in 2016, proposed a methodology of using a Random decision forests [37]. These forests were used to automatically detect cracks within concrete structures. The concrete structures were represented as, images that contained a concrete crack or not. This forms two classes, where the images can be classified into. This is known as a Binary Classification problem. The objective of the research was to solve this particular form of binary classification. This was done by training and testing on a combination of AigleRN and CFD. The results achieved were, a Continuity Index (CI) of 0.87 and 0.67, respectively.

However, civil infrastructure damage may include non-crack damages. For example, spallation, efflorescence, exposed bars, corrosion (stains). These damages have not been addressed in the image dataset. Additionallly, the images were taken in ideal illumination and minimal background-related information. This is an inherent limitation of the study. This limitation means that, the defect classifier has not been trained to classify non-crack damages, in suboptimal image capture environments. As a result, the classifier may not be widely used in practice.

Research conducted by Yang et. all in 2017, proposed a methodology of using finely-tuned VGG16 [48]. This CNN would be used, in conjunction with Simultaneous Localisation and Mapping (SLAM), to automatically locate concrete spallings and cracks. These types of damages were represented as, 278 and 954 images, respectively. The classifier achieved a test accuracy of 93%.

However, civil infrastructure damage may include efflorescence, exposed bars and corrosion (stains). These damages have not been addressed in the image dataset, which is an inherent limitation of the study. This limitation means that, the defect classifier has not been trained to classify non-crack and non-spalling damages. As a result, the classifier received an field test accuracy of 70%, therefore the error rate is 30%. This error rate may be considered too high, for the classifier to be used in practice.

Research conducted by Silva et. all in 2018, proposed a methodology of using a VGGNet [38]. This CNN was used to detect cracks within concrete structures. The concrete structures were represented as, images that contained a concrete crack (i.e. 2336 images) or not (i.e. 1164 images). This forms two classes, where the images can be classified into. This is known as a Binary Classification problem. The objective of the research was to solve this particular form of binary classification. This was done by training and testing the VGGNet on the images. The model achieved a test accuracy of 92.27%.

However, the limitations of this research parallel that of "Automatic Road Crack Detection Using Random Structured Forests" [37].

This trend is prevalent in subsequent research, where CNN architectures have been trained and its performance evaluated through testing [25, 12, 30, 11, 24]. Unfortunately, the limitation stated is also prevalent.

However, the research conducted by Wang et. all uses AlexNet and GoogleLeNet. These CNN architectures were trained to identify the multidamage (i.e. spallation, crack, efflorescence and intact) with an image dataset of the Forbidden City Wall in China. The entire dataset consisted of small dataset and a large dataset. The small dataset consisted of 2,000 training images and 400 validation and testing images. The large dataset consisted of 20,000 training images and 4,000 validation and testing images. The testing accuracy was 94.3%. This is an instance of CNNs being used to perform multi-class classification.

The image dataset consists of a higher range of common civil infrastructure defects. This allows for the classifiers to be trained on classifying such defects, thus reduces the

errors that are associated with the trending limitations of past research.

This concept was further expanding in the research conducted by, Mundt et. al in 2019. The research introduced the COncrete DEfect BRidge IMage (CODEBRIM) image dataset [34]. This image dataset contains the cropped patches with corresponding class labels split into training (4296 images), validation (466 images) and test (482 images) sets for machine learning.

The research used Meta-QNN and ENAS classifiers. These classifiers were trained and their performance evaluated using the CODEBRIM dataset. The results achieved were, validation accuracies of upto 66% and test accuracies of upto 72%.

Defect Database (Year)	Database Constituents	Research Objective	CNN used	Results
CFD, AigleRN (2016) [37]	118 RGB images with a resolution of 320 X 480 pixels and 38 gray-level images, respectively	Pavement crack detection via. 60% 40% training/testing split with the images reduced to 480 X 320	Random decision forests	Train/ Test: Continuity Index (CI) = 0.87 (AigleRN/ AigleRN), 0.67 (CFD/ CFD)
Concrete Structure Spalling and Crack (CSSC) (2017) [48]	Concrete spalling: 278 images for training and Concrete crack: 954 images for crack detection	Automatic locating of concrete spalling and cracks via. UAV and Simultaneous Localisation and Mapping (SLAM)	Fine-Tuned VGG-16 (FVGG)	Test accuracy: 93%, Field test accuracy = 70%
n/a (2018) [38]	3500 total images (Concrete cracks (2336) and concrete without cracks (1164))	Develop a machine learning-based model to detect cracks on concrete surfaces	VGG16	Test accuracy: 92.27%
n/a (2018) [25]	487 images, 3186 Crack Candidate Region (CCR), Concrete cracks (527) and concrete without cracks (2659))	Image classification for concrete cracks via. CNN	AlexNet	Validation accuracy: 98%
SDNET2018 (2018) [12]	56,000 images of with a variety of obstructions, including shadows, surface roughness, scaling, edges, holes, and background debris cracked (narrow as 0.06 mm and as wide as 25 mm, 230 images) and non-cracked concrete bridge decks, walls, and pavements	Image classification for concrete cracks via. CNN	AlexNet	Average Testing accuracy ((Bridge deck): 91.18%,(Wall): 88.42%, (Pavement): 95.19%
n/a (2018) [11]	19 high definition images (3420 sub-images, 319 with cracks and 3101 without)	Image-based crack detection in concrete structures via. DCNN and edge detection	AlexNet, Transfer learning	Testing accuracy: 99%
n/a (2018) [30]	45 images for training (56,000 positive samples and 270,000 negative samples), 5 images for testing	detect cracks in a pixel-wise manner from real concrete surface images	AlexNet, Local Pattern Predictor (LPP), Spatially tuned robust multifeature (STRUM) classifier	Testing accuracy: 99%
n/a (2018) [24]	Concrete crack images (2073), Joint (1400), Plant (1511), Intact (2211)	Crack detection with images scraped from the Internet	AlexNet, Transfer learning	Testing accuracy: 96.64%
n/a (2018) [46]	Images of the Forbidden City Wall in China: Small dataset (2,000 images for training, 400 images for validation and testing) and a large dataset (20,000 images for training, 4,000 images for validation and testing)	Identify the multidamage (spall, crack, efflorescence, intact) of historic masonry structures based on CNN techniques	AlexNet, GoogLeNet	Testing accuracy: 94.3%
n/a (2019) [29]	Trained and validated using a built database with 60000 images	Use CNN for crack detection	(Modified) AlexNet	Validation accuracy: 99.06%
COncrete DEfect BRidge IMage (CODEBRIM) (2019) [34]	1590 total images from 30 unique bridges (5354 defect bounding boxes, 2506 generated non-overlapping background bounding boxes)	Use CNN for multi-target classification of five commonly appearing concrete defects	Meta-QNN, ENAS	Validation accuracy: 99.06%

Table 3: Research Summary of Automatic defect detection, which highlights the database constituents, CNN used for classification and CNN results.

Aims and Objectives

- Provide an answer to the research question: What is the current SOTA Inception-based CNN architectures' damage classification performance on a multi-target, multi-class concrete defect image dataset?
- To use transfer learning to implement InceptionV3, InceptionV4, InceptionResNetV2 and Xception, (all with justified parameters) for CODEBRIM defect classification.
- To evaluate CNN performance using Accuracy, Precision, Recall, F1-Score and Specificity.
- Use convergence studies to gain insight into the current SOTA Inception-based CNNs' defect classification performance.
- To produce a comparison of popular non-Inception CNNs with SOTA Inception-based CNNs.
- To identify the limitations of current SOTA Inception-based CNNs.

Chapter 3

Implementation and Development Methodology

3.1 Database

Figure 5: (a) Top row from left to right: 1.) exposed bars, spallation, cracks (hard to see) 2.) hairline crack with efflorescence 3.) efflorescence 4.) defect-free concrete. Bottom row from left to right: 1.) large spalled area with exposed bars and corrosion 2.) crack with graffiti 3.) corrosion stain, minor onset efflorescence 4.) defect-free concrete with dirt and markings.

Figure 6: (b) From left to right: 1.) spalled area with exposed bar, advanced corrosion and efflorescence 2.) exposed corroded bar 3.) larger crack 4.) partially exposed corroded bars, cracks 5.) hairline crack 6.) heavy spallation, exposed bars, corrosion 7.) wet/damp crack with efflorescence on the top 8.) efflorescence 9.) spalled area 10.) hairline crack with efflorescence.

The largest and most challenging bridge defect database is CODEBRIM, which is comprised of 30 unique bridges (i.e. 150 randomly chosen unique defect examples per class for validation and test sets respectively). These images have been acquired by a UAV and four different high resolution cameras, in varying distance, angles, illumination and weather conditions. Each of the 30 bridges varied in texture, structure and the number of each defect. Specifically, there are 2507 crack images, 1898 spallation images, 833 efflorescence images, 1507 exposed bars images and 1559 corrosion stain images.

The CODEBRIM image dataset has unique properties such as, any image within the dataset may display more than one type of damage. The reason for this is that, the types of defects can overlap within an image. As a result, an image may be classified into multiple classes. This strives to replicate the appearance of concrete material (i.e. stains, holes, colors, poster remains, graffiti, marking and painting), illumination and weather conditions, in practice. See Figures 5 and 6. This is a multi-target, multi-class classification problem.

Civil infrastructure damage may include stains, holes, colors, poster remains, graffiti, marking and painting. These damages have been accounted for within the image dataset. This is not true for civil infrastructure image datasets of the past. As a result, defect classification is of a higher difficulty, when compared to past civil infrastructure image datasets. Any defect classifier, that has been trained to classify common civil defects automatically using the CODEBRIM image dataset, is likely to result in being

more readily trained for practical use, than similar past classifiers.

Given the inherent characteristics of the CODEBRIM image classification dataset explained above, this dataset has been used for this thesis. The original CODEBRIM paper performed defect classification using non-Inception CNN architectures (See figure 19). As a result, the knowledge of how SOTA Inception CNN architectures would perform on the CODEBRIM image dataset is unknown. By obtaining this knowledge, insight into the limitations of SOTA Inception CNN architectures may be revealed. If this is true, then these insights may be used to improve CNN architecture design in future image classification works.

3.2 Data Visualisation

Once the images were captured they underwent an bounding-box-based annotation process (See Figure 7. This process involved removing non-defective images and containing each defect within the Pascal format [13]. Once this has been completed, each bounding block from every class is assigned a label. The label is dependent on whether a defect is present or not. Finally, a sample is taken. The sample contains bounding boxes with concrete and non-concrete defects. These are sampled based on specified parameters (e.g. fixed count). This process resulted in different defects, within an image, being separated and labelled.

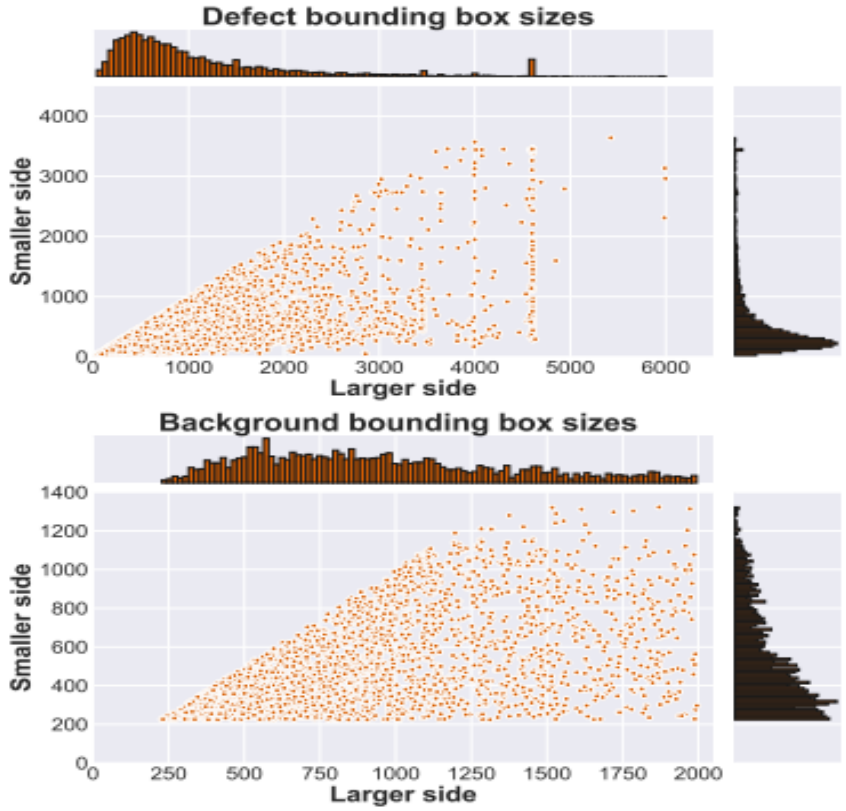


Figure 7: Top graph: Distribution of annotated bounding box sizes for defects. Bottom panel: distribution of sizes for sampled non-overlapping background bounding boxes [34].

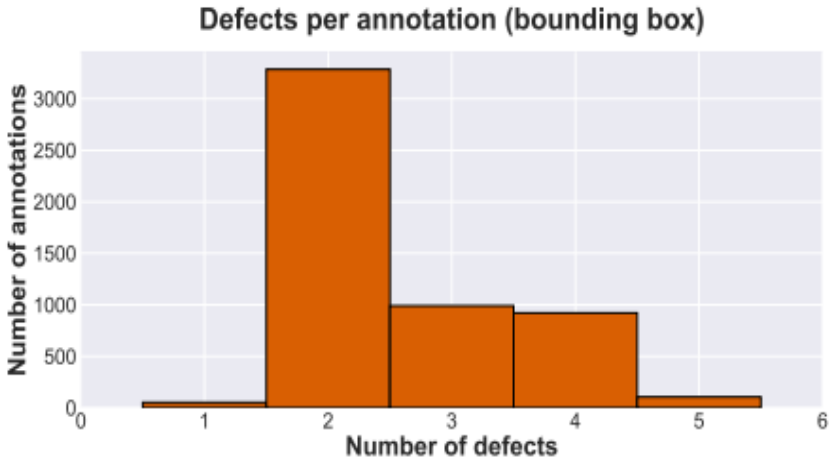


Figure 8: Histogram of number of simultaneously occurring defect classes per annotated bounding box [34].

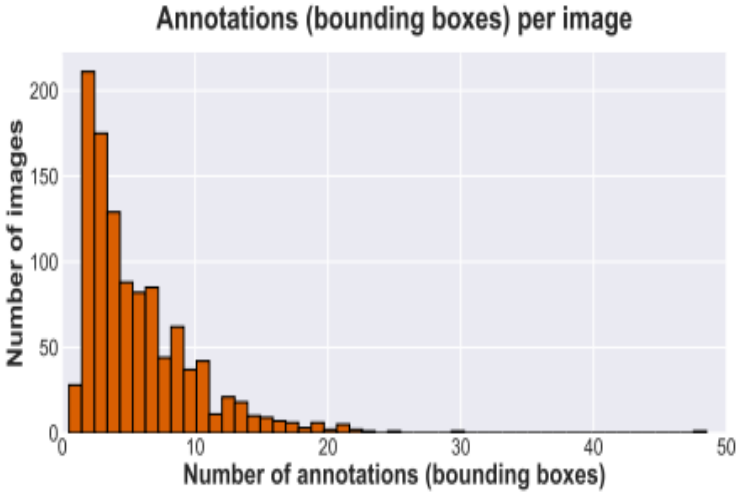


Figure 9: Distribution of number of bounding box annotations per image [34].

Figures 9 and 8 show that, due to the 30 selected bridges, most images used contain multiple different defect locations.

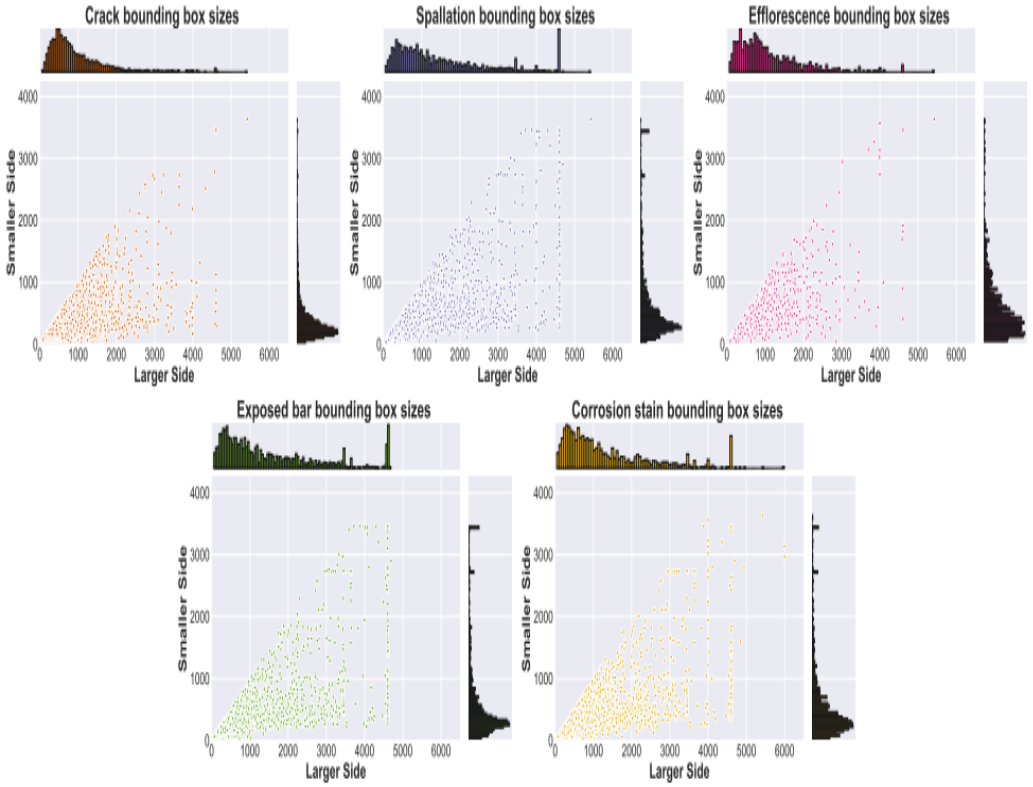


Figure 10: Individual distributions of annotated bounding box sizes for each of the 5 defect classes [34].

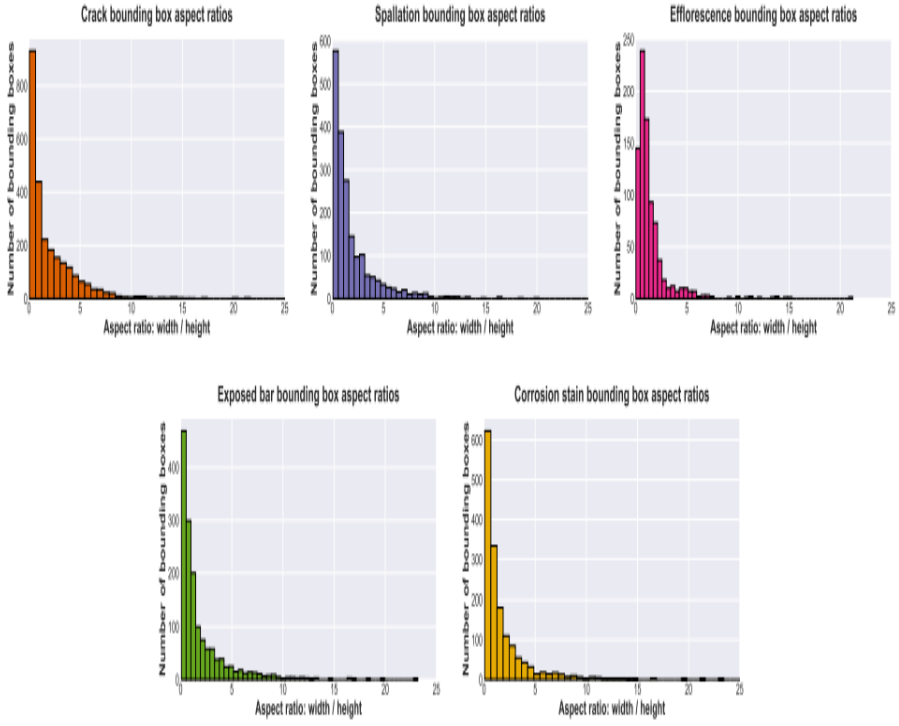


Figure 11: Individual distributions of number of bounding box annotations for different aspect ratios for each of the 5 defect classes [34].

Figures 10 and 11 show that, the bounding boxes have multiple areas of interest that overlap. This causes the distribution per class to not be mutually exclusive. Additionally, figure 11 shows that, the difference between the bounding box sizes for each class is not large. Furthermore, the bounding box aspect ratio for cracks is the highest, efflorescence is the lowest and all the rest are similar to one other.

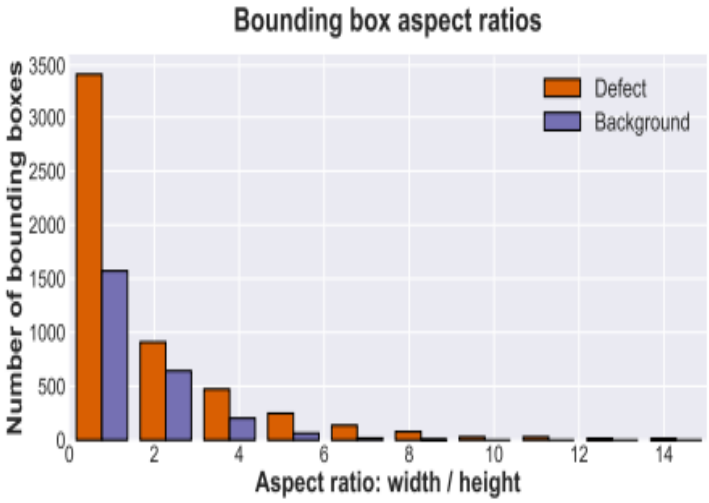


Figure 12: Distribution of number of defect and background bounding box annotations for different aspect ratios [34].

3.3 Data Pre-processing

(a) Example Background XML

(b) Example Background Camera

(c) Example Defect Camera

(d) Example Defect XML

Figure 13: CODEBRIM dataset (original format)

Once the CODEBRIM image dataset (classification version) has been downloaded from the internet [33], the data is formatted as shown in Figure 13. The figure shows that, the CODEBRIM dataset is structured as a pairing of XML and the high-resolution image. The pairing exists for background and defect data. The XML file includes annotation data for each defect class, for each of the images. This is expressed in binary (i.e. 0 for False, 1 for True). Furthermore, the pairings have been organised in separate folders for training, testing and validation.

The CODEBRIM dataset has overlapping defects in a single image. To account for these defects, all nodes in the output layer of the Inception-based CNN architecture represent a defect per node and ultilise a Sigmoid Activation function. This function

results in binary decision being made per output node (i.e. If defect is present, then 0. If defect is not present, then 1). This has resulted in Binary Cross entropy loss function being used.

```
...
[0.07058824 0.06666667 0.05098039]
[0.07058824 0.06666667 0.05098039]
[0.06666667 0.0627451  0.04705882]]
```

Figure 14: Normalised numerical matrix-image representation example

The pre-processing steps taken were, to use iteration loops and arrays to resize each image to a specified size, store the numerical matrix-image representation and their corresponding defect class label. The Sigmoid Activation function range of outputs are from -1 to +1, therefore the numerical matrix-image representation must be divided by 255. This acts as a normalisation function.

```
[179. 167. 151.]
[173. 164. 147.]
[171. 164. 148.]]
```

Figure 15: Unormalised numerical matrix-image representation example

If these inputs were not normalised, then the cost function becomes elongated. This increases the difficulty in cost function optimisation, thereby lessening the training time, when compared to its normalised counterpart [4].

3.4 Preparing the CNN

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks [16]. Tensorflow includes Keras applications. These applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning [45].

InceptionV3, Xception and InceptionResNetV2 are Keras applications. InceptionV4 has been downloaded from another online repository [23] (See figures 20, 22, 17 and 21, respectively). InceptionV4 has more layers and branches compared to InceptionV3. InceptionV4 does not include Residual Networks and InceptionResNetV2 does. These Inception networks are depth-based and Xception is width-based. All models have been instantiated using the Keras applications and have had additional layers appended to them for the required task (e.g. Dense layer with Sigmoid activation function) (i.e. transfer learning).

3.5 CNN Model Training, Testing and Validation

A CNN architecture must be trained to correctly classify the defects found in an image. The training, validation and testing procedures are the same as the procedures from the original CODEBRIM paper [34]. The procedure begins by using a Sigmoid activation function, with a threshold of 0.5, in conjunction with the Binary cross entropy loss function.

The Matthews Correlation Coefficient (MCC) (see equation A.1) is used to measure of the quality of binary and multi class classifications. In order to do this, true and false positives and negatives are considered independent of class sizes. The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction [8].

The MCC used in conjunction with a Sigmoid Activation function allows for the threshold of the activation function to be known. These knowledge would allow for the class labels to also become known. If the actual class label and predicted class labels are the same, then a correct defect classification has been made.

Secondly, 150 background images and 150 unique defect images were chosen for each class. This was done for the validation and test sets. The defects and backgrounds of different images may be similar, with the exception of image resolution and size. This may result in overfitting. To reduce the probability of overfitting, a single image's annotated bounding boxes are included in only one dataset split. To remedy the varying

image sizes and scales, each image patch is rescaled to a pre-determined patch size. The research conducted by Gaurab Bhattacharya trained a CNN classifier on patch sizes of 96, 128, 160 and 192 pixels and batch sizes of 16 and 32 images [6]. A Stochastic Gradient Descent optimiser was used. Figure 4 shows that, the highest training, validation and test accuracy was achieved by a batch size of 32 images and a patch size of 96 pixels. This parameters were used in this dissertation, alongside a learning rate of 0.01, weight decay of 10^{-6} and momentum as 0.9.

Patch	Batch size: 16			Batch size: 32		
	Training accuracy	Validation accuracy	Test accuracy	Training accuracy	Validation accuracy	Test accuracy
96	**99.91**	**84.58**	**82.79**	99.92	80.56	79.24
128	99.89	82.33	79.56	99.84	79.55	78.26
160	99.85	80.06	78.85	99.58	80.56	76.85
192	99.78	79.56	78.26	99.69	79.78	75.38

Table 4: Comparison of multi-target validation accuracy and best validation model and test accuracy based on a patch-batch combinations [6]

.

The number of images per class within the training data is not the same. This may lead to a CNN, which can correctly classify only those classes, which are not under-represented. To counteract this training concern, the under-represented class examples are virtually replicated. This equally represents all classes during training. The test and validation datasets are already balanced.

The validation and testing protocol used, follows the protocol used in the original CODEBRIM paper [34] Once the CNN has been successfully trained, the validation and testing datasets are used for validation and testing the performance of the CNN, respectively.

Chapter 4

Experimental Results

4.1 Performance Metrics

A confusion or error matrix is a formatted table. This table is used to visualise the performance of an algorithm [40]. This is done by calculating four aspects. These aspects are known as, True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). A TP occurs when the model correctly predicts the positive class. A TN occurs when the model correctly predicts the negative class. A FP occurs when the model incorrectly predicts the positive class. A FN occurs when the model incorrectly predicts the negative class.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

The accuracy expresses the number of correct classifications during cross-validation.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

The precision expresses the accuracy of positive prediction only.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

The recall or sensitivity expresses the ratio of positive instances that are correctly

detected by the classifier.

$$Specificity = \frac{TN}{TN + FP} \tag{4.4}$$

The specificity expresses the ratio of negative instances that are correctly classified as negative.

$$F1 - Score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{4.5}$$

The F1-Score is a harmonic mean of precision and recall.

4.2 Experimental Results

Performance Metric	Crack	Spallation	Efflorescence	Exposed Bar	Corrosion
Accuracy	0.9253	0.8963	0.9191	**0.944**	0.8755
Precision	0.875	0.8425	**0.9297**	0.902	0.7647
Recall	0.8867	0.82	0.7987	**0.92**	0.8667
F1-Score	0.8808	0.8311	0.8592	**0.9109**	0.8125
Specificity	0.9428	0.9307	**0.973**	0.9548	0.8795

Table 5: Inception v3 Results. The highest scores are in bold. The Exposed Bar had the highest Accuracy, Recall and F1-Scores, whilst the Efflorescence class had the highest Precision and Specificity, but the lowest Recall score (i.e. the Efflorescence class had the worst ratio of positive instances that are correctly detected by the classifier, when compared to all other defect classes).

Performance Metric	Crack	Spallation	Efflorescence	Exposed Bar	Corrosion
Accuracy	0.7531	0.8527	0.8361	**0.9191**	0.7469
Precision	0.5683	0.797	0.8241	**0.8881**	0.675
Recall	**0.86**	0.7067	0.5973	0.8467	0.36
F1-Score	0.6844	0.7491	0.6926	**0.8669**	0.4696
Specificity	0.7048	0.9187	0.9429	**0.9518**	0.9217

Table 6: Inception v4 Results. The highest scores are in bold. The Exposed Bar had the highest Accuracy, Precision, Specificity and F1-Scores, whilst the Crack class had the highest Recall. The Corrosion class received the lowest Recall score (i.e. the Corrosion class had the worst ratio of positive instances that are correctly detected by the classifier, when compared to all other defect classes).

Performance Metric	Crack	Spallation	Efflorescence	Exposed Bar	Corrosion
Accuracy	0.917	0.8921	0.9046	**0.9336**	0.8714
Precision	0.8438	0.8182	0.8652	**0.9758**	0.7418
Recall	**0.9**	0.84	0.8188	0.8067	**0.9**
F1-Score	0.871	0.8289	0.8414	**0.8832**	0.8133
Specificity	0.9247	0.9157	0.9429	**0.991**	0.8584

Table 7: InceptionResNetV2 Results. The highest scores are in bold. The Exposed Bar had the highest Accuracy, Precision, Specificity and F1-Scores, whilst the Crack and Corrosion classes had the highest Recall. The Corrosion class also received the lowest Precision score (i.e. the accuracy of positive predictions of defects within the Corrosion class is the lowest compared to all other defect classes).

Performance Metric	Crack	Spallation	Efflorescence	Exposed Bar	Corrosion
Accuracy	0.9315	0.8631	0.9108	**0.9378**	0.8714
Precision	0.9149	0.7333	0.8841	**0.9286**	0.7444
Recall	0.86	0.88	0.8188	0.8667	**0.8933**
F1-Score	0.8866	0.8	0.8502	**0.8966**	0.8121
Specificity	0.9639	0.8554	0.952	**0.9699**	0.8614

Table 8: Xception Results. The highest scores are in bold. The Exposed Bar had the highest Accuracy, Precision, Specificity and F1-Scores, whilst the Corrosion class had the highest Recall. The Spallation class also received the lowest Precision score (i.e. the accuracy of positive predictions of defects within the Spallation class is the lowest compared to all other defect classes).

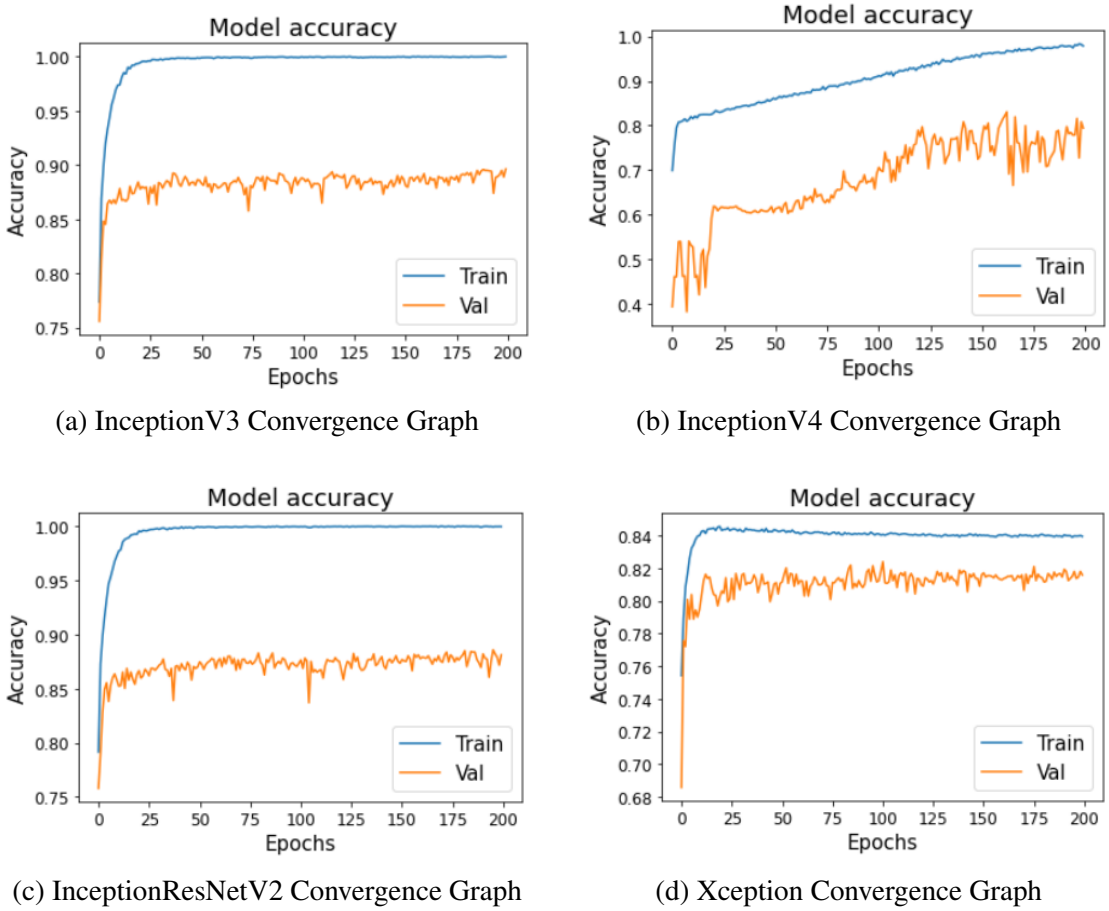
CNN Architecture	Test Accuracy
InceptionV3	0.91
InceptionV4	0.74
InceptionResNetV2	0.87
Xception	0.89

Table 9: Test accuracy of the four SOTA inception-based CNN architectures used to classify multi-class, multi-target of five frequently occurring concrete defects.

CNN Architecture	Multi-target accuracy [%]		Params [M]
	Best Validation	Best Validation - Test	
AlexNet	63.05	66.98	57.02
T-CNN	64.3	67.93	58.6
VGG-A	64.93	70.45	128.79
VGG-D	64.0	70.61	134.28
WRN-28-4	52.51	57.19	5.84
Densenet-121	65.56	70.77	11.5
ENAS-1	65.47	70.78	3.41
ENAS-2	64.53	68.91	2.71
ENAS-3	64.38	68.75	1.70
MetaQNN-1	66.02	68.56	4.53
MetaQNN-2	65.20	67.45	1.22
MetaQNN-3	64.93	72.19	2.88
InceptionV3	89.61	**91.0**	21.8
InceptionV4	79.44	74.0	41.1
InceptionResNetV2	88.11	87.0	54.4
Xception	81.59	89.0	21.2

Table 10: Comparative performance analysis of the four SOTA Inception-based CNN architectures (below bold line) with other SOTA CNN architectures (above the line) in CODEBRIM defect classification (i.e. multi-class, multi-target classification). The CNN Architecture (i.e. InceptionV3) and its associated highest test accuracy is highlighted in bold.

4.3 Discussion



(a) InceptionV3 Convergence Graph

(b) InceptionV4 Convergence Graph

(c) InceptionResNetV2 Convergence Graph

(d) Xception Convergence Graph

Figure 16: Convergence curves produced during training and validation using four SOTA Inception-based CNN architectures, namely InceptionV3, InceptionV4, InceptionResNetV2 and Xception, respectively, on the CODEBRIM dataset. The blue and orange curves represent the training and validation accuracies, respectively. These are represented over the number of epochs (i.e 200 epochs). InceptionV3, InceptionResNetV2 and Xception's convergence curves demonstrate that 200 epochs is sufficient for successful convergence. InceptionV4's convergence curve demonstrates that 200 epochs is not sufficient for successful convergence, therefore more than 200 epochs are required for the successful convergence of the InceptionV4 model.

A graph of convergence shows, the correlation between the model's accuracy and the number of epochs. This type of graph has been to review a CNN model's performance. The InceptionV4 convergence graph (see figure 16b) shows, the training and validation accuracy curves not becoming asymptotic as the number of epochs increases, convergence did not occur. The training accuracy rapidly increases, in the initial epochs, followed by a gradual improvement. Between 160 to 173 epochs, there is a large decrease in validation accuracy, as the training accuracy continues to improve. This is indicative that, the number of model parameters is too large to give the optimal prediction, and is

unable to generalised to the testing dataset (i.e. overfitting is the cause of the lowest test accuracy by a minimum of 13%).

All four SOTA inception-based CNN architectures have greater best validation and best validation on the test dataset, than popular CNNs from the literature. InceptionV3 has the highest best validation and best validation on the test dataset, than the other inception-based CNN architectures used. This contradicts the trend that, more complex, younger CNN architectures outperform their predecessors.

Chapter 5

Conclusion

5.1 Conclusion and Future Works

In this work, the challenges that have been addressed are of classification of defect images within the CODEBRIM dataset and implements a solution for automatic detection and classification of bridge defects. The implementation of Inception-based CNN architectures allow for the extraction of robust, salient concrete defects, which have been classified into five common defect classes. Experimental results and convergence studies show how the implemented Inception-based CNN architectures outperforms the other popular SOTA CNN architectures. Specifically, for the challenging multi-class classification problem, InceptionV3 obtained the highest testing accuracy of 91.0%, compared to 72.19% by the MetaQNN-3 CNN architecture (from the original CODEBRIM research). Although, InceptionV4 is a more complex, updated version of InceptionV3, its testing accuracy of only 74.0%, is lower than its predecessor due to overfitting.

Future work will perhaps involve identifying the limitations of current SOTA Inception-based CNNs and improving CNN architecture performance. Genetic algorithms have proven effective in designing and parameter optimisation [41]. Efforts have been made to try to adapt such a CNN genetic algorithm for the multi-class classification problem. Consequently, due to lack of time, this dissertation does not cover such work. However, these efforts are still being continued as such adaptions may result in improved CNN

performance via. the genetic algorithm's fitness function being based on classification accuracy of an individual (i.e. CNN architecture).

The contributions of this dissertation have defined the performance of the current SOTA Inception-based CNNs on the CODEBRIM image dataset.This sets the benchmark for future classifiers to strive to improve and overcome the challenge of multi-class defect classification.

Appendix A

Appendix

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{A.1}$$

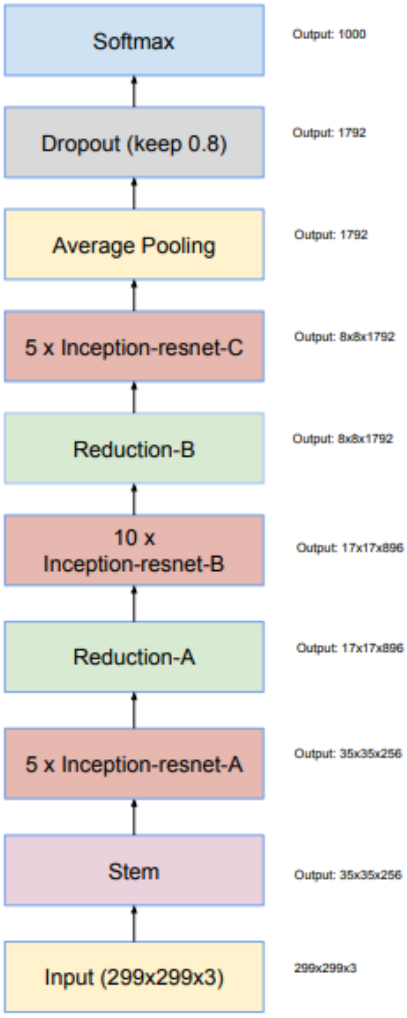


Figure 17: The overall schema of the Inception-ResNet-v1 and InceptionResNet-v2 networks.

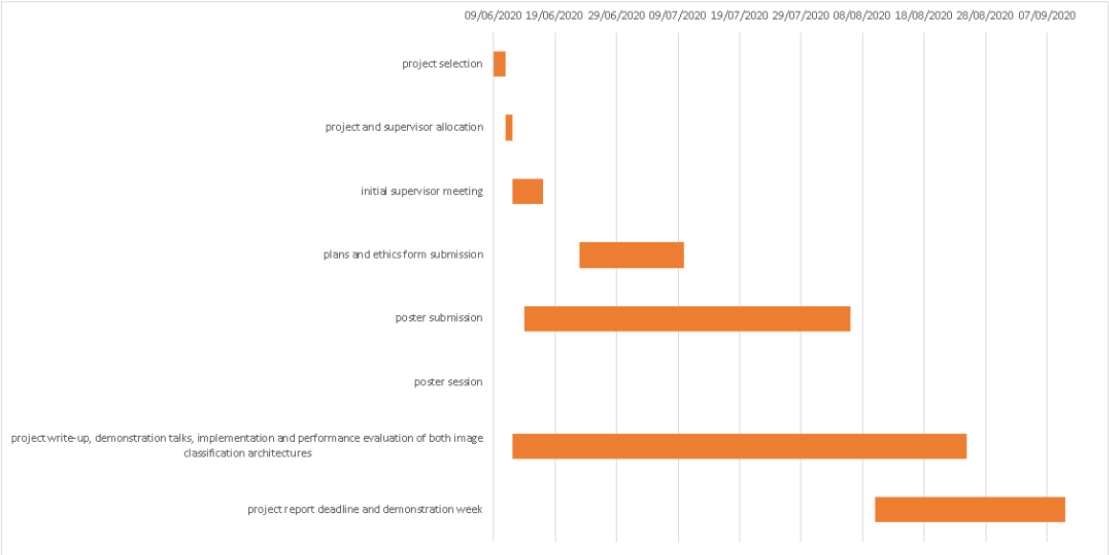


Figure 18: MSc Project Gaant Chart.

Architecture	Multi-target accuracy [%]		Params [M]	Layers
	best val	bv-test		
Alexnet	63.05	66.98	57.02	8
T-CNN	64.30	67.93	58.60	8
VGG-A	64.93	70.45	128.79	11
VGG-D	64.00	70.61	134.28	16
WRN-28-4	52.51	57.19	5.84	28
Densenet-121	65.56	70.77	11.50	121
ENAS-1	65.47	70.78	3.41	8
ENAS-2	64.53	68.91	2.71	8
ENAS-3	64.38	68.75	1.70	8
MetaQNN-1	66.02	68.56	4.53	6
MetaQNN-2	65.20	67.45	1.22	8
MetaQNN-3	64.93	72.19	2.88	7

Figure 19: Comparison of popular CNNs from the literature with the top three architectures of MetaQNN and ENAS in terms of best multi-target validation accuracy (best val), best validation model's test accuracies (bv-test), overall amount of parameters (Params) in million and amount of trainable layers. For WRN we use a width factor of 4 and a growth rate of k = 32 for DenseNet

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 20: InceptionV3 Architecture

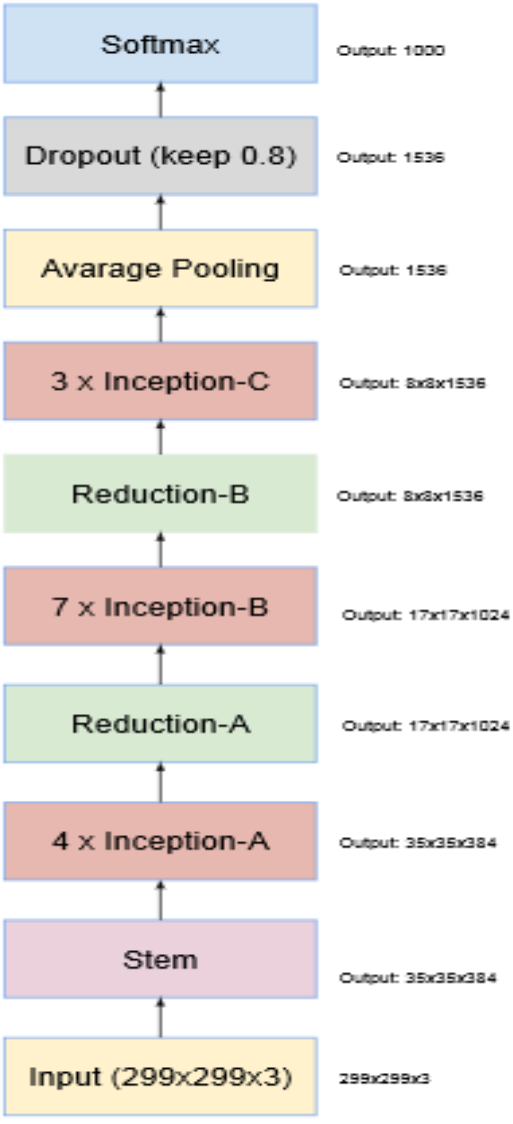


Figure 21: InceptionV4 Architecture

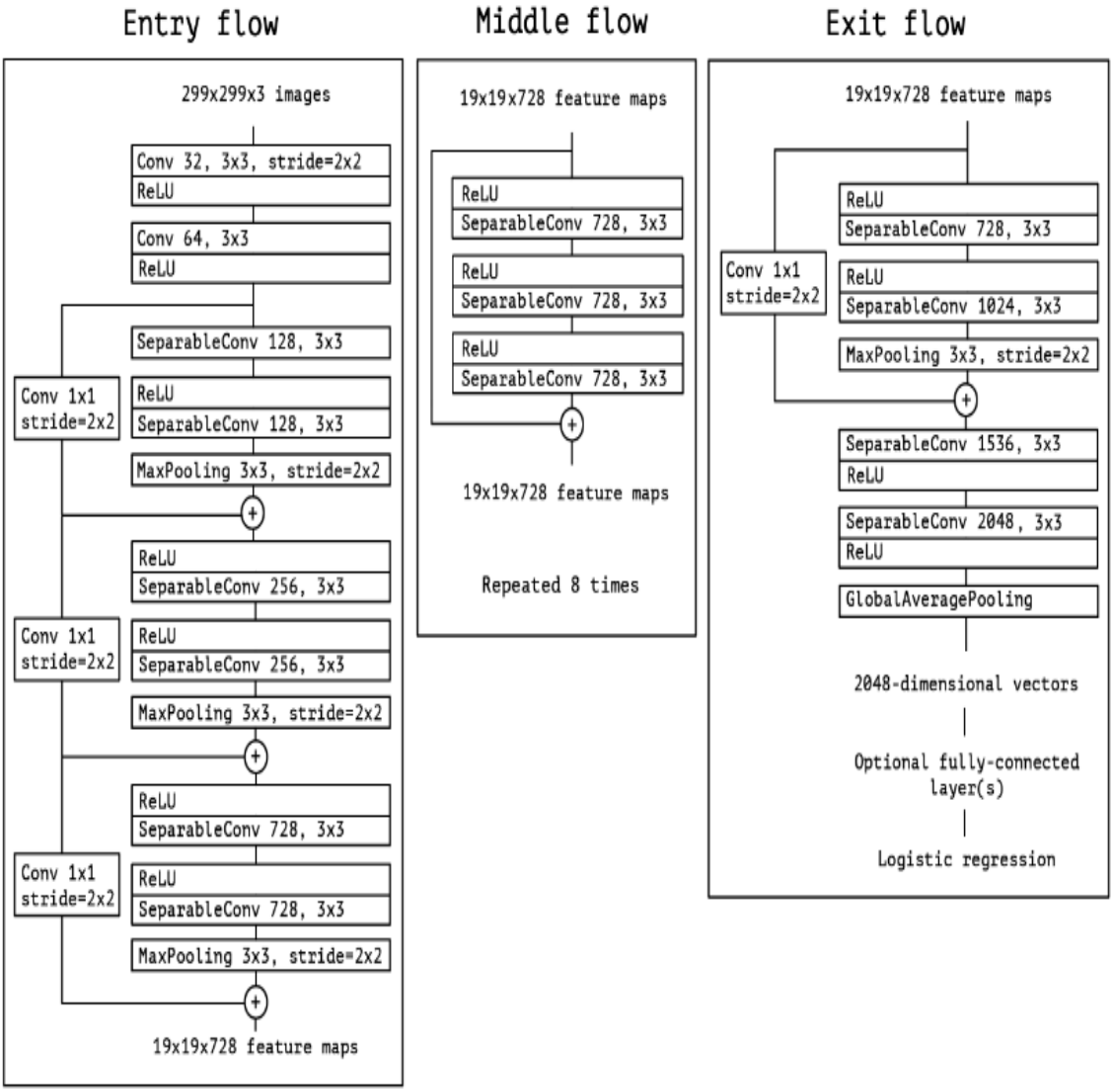


Figure 22: Xception Architecture

List of Figures

Bibliography

[1] URL: http://www.pcrichardson.co.uk/bridge-maintenance.

[2] Sami Abu-El-Haija et al. "YouTube-8M: A Large-Scale Video Classification Benchmark". In: *CoRR* abs/1609.08675 (2016). arXiv: 1609.08675. URL: http://arxiv.org/abs/1609.08675.

[3] Arcadis. *Result-oriented asset management*. URL: https://www.arcadis.com/en/united-kingdom/what-we-do/our-projects/europe/netherlands/result-oriented-asset-management/.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[5] Y. Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 5 (Feb. 1994), pp. 157–66. DOI: 10.1109/72.279181.

[6] Gaurab Bhattacharya. "Attention-based deep learning formulti-target multi-class structural defectclassification". PhD thesis. 2020.

[7] L. Chen et al. "Online multi-object tracking with convolutional neural networks". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 645–649.

[8] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evalu-

ation". In: *BMC Genomics* 21 (Dec. 2020). DOI: 10.1186/s12864-019-6413-7.

[9] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *CoRR* abs/1610.02357 (2016). arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357.

[10] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[11] Sattar Dorafshan, Robert Thomas, and Marc Maguire. "Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete". In: *Construction and Building Materials* 186 (Aug. 2018), pp. 1031–1045. DOI: 10.1016/j.conbuildmat.2018.08.011.

[12] Sattar Dorafshan, Robert Thomas, and Marc Maguire. "SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks". In: *Data in Brief* 21 (Nov. 2018). DOI: 10.1016/j.dib.2018.11.015.

[13] Mark Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". English. In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136. ISSN: 0920-5691. DOI: 10.1007/s11263-014-0733-5.

[14] Wulfram Gerstner and Werner Kistler. "Mathematical Formulations of Hebbian Learning". In: *Biological cybernetics* 87 (Jan. 2003), pp. 404–15. DOI: 10.1007/s00422-002-0353-y.

[15] Xavier Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Journal of Machine Learning Research - Proceedings Track* 9 (Jan. 2010), pp. 249–256.

[16] Google. *TensorFlow: Open source machine learning*. URL: https://www.youtube.com/watch?v=oZikw5k_2FM.

[17] *Hammersmith Bridge closed because cracks in pedestals*. May 2019. URL: https://www.bbc.co.uk/news/uk-england-london-48395371.

[18] Dongyoon Han, Jiwhan Kim, and Junmo Kim. “Deep Pyramidal Residual Networks”. In: *CoRR* abs/1610.02915 (2016). arXiv: 1610.02915. URL: http://arxiv.org/abs/1610.02915.

[19] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[20] Kaiming He et al. “Mask R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[21] Rob Horgan. *Fatal Taiwan bridge collapse is latest example of maintenance failings*. Oct. 2019. URL: https://www.newcivilengineer.com/latest/fatal-taiwan-bridge-collapse-is-latest-example-of-maintenance-failings-07-10-2019/.

[22] *Italy bridge collapse: Genoa death toll rises to 43*. Aug. 2018. URL: https://www.bbc.co.uk/news/world-europe-45241842.

[23] Kentsommer. *kentsommer/keras-inceptionV4*. URL: https://github.com/kentsommer/keras-inceptionV4.

[24] Byunghyun Kim and Soojin Cho. “Automated Vision-Based Detection of Cracks on Concrete Surfaces Using a Deep Learning Technique”. In: *Sensors* 18 (Oct. 2018). DOI: 10.3390/s18103452.

[25] Hyunjun Kim et al. “Crack and Noncrack Classification from Concrete Surface Images Using Machine Learning”. In: *Structural Health Monitoring* (Apr. 2018), p. 147592171876874. DOI: 10.1177/1475921718768747.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[27] *Large Scale Visual Recognition Challenge 2017 (ILSVRC2017)*. URL: http://image-net.org/challenges/LSVRC/2017/index.

[28] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551.

[29] Shengyuan Li and Xuefeng Zhao. "Image-Based Concrete Crack Detection Using Convolutional Neural Network and Exhaustive Search Technique". In: *Advances in Civil Engineering* 2019 (Apr. 2019), pp. 1–12. DOI: 10.1155/2019/6520620.

[30] Yundong Li, Hongguang Li, and Hongren Wang. "Pixel-Wise Crack Detection Using Deep Local Pattern Predictor for Robot Application". In: *Sensors* 18 (Sept. 2018), p. 3042. DOI: 10.3390/s18093042.

[31] *Local highways maintenance challenge fund: schemes funded*. URL: https://assets.publishing.service.gov.uk/government/%5C%5Cuploads/system/uploads/attachment%5C_data/file/869349/challenge-fund-schemes.csv/preview.

[32] D. Mehta et al. "Monocular 3D Human Pose Estimation in the Wild Using Improved CNN Supervision". In: *2017 International Conference on 3D Vision (3DV)*. 2017, pp. 506–516.

[33] Martin Mundt et al. *CODEBRIM: COncrete DEfect BRidge IMage Dataset*. Apr. 2019. URL: https://zenodo.org/record/2620293.

[34] Martin Mundt et al. "Meta-learning Convolutional Neural Architectures for Multi-target Concrete Defect Classification with the COncrete DEfect BRidge IMage Dataset". In: *CoRR* abs/1904.08486 (2019). arXiv: 1904.08486. URL: http://arxiv.org/abs/1904.08486.

[35] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 91–99. URL:

http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[36] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[37] Y. Shi et al. “Automatic Road Crack Detection Using Random Structured Forests”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.12 (2016), pp. 3434–3445.

[38] Wilson Silva and Diogo Schwerz de Lucena. “Concrete Cracks Detection Based on Deep Learning Image Classification”. In: vol. 2. June 2018, p. 5387. DOI: 10.3390/ICEM18-05387.

[39] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv 1409.1556* (Sept. 2014).

[40] Stephen Stehman. “Selecting and interpreting measures of thematic classification accuracy”. In: *Remote Sensing of Environment* 62 (Oct. 1997), pp. 77–89. DOI: 10.1016/S0034-4257(97)00083-7.

[41] Y. Sun et al. “Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification”. In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3840–3854.

[42] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”. In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

[43] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[44] Christian Szegedy et al. “Rethinking the Inception Architecture for Computer Vision”. In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[45] Keras Team. *Keras documentation: Keras Applications*. URL: https://keras.io/api/applications/#:~:text=Keras%20Applications%20are%20deep%20learning.

[46] Niannian Wang et al. "Damage Classification for Masonry Historic Structures Using Convolutional Neural Networks Based on Still Images: Damage classification for masonry historic structures using CNNs". In: *Computer-Aided Civil and Infrastructure Engineering* (Aug. 2018). DOI: 10.1111/mice.12411.

[47] Saining Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". In: *CoRR* abs/1611.05431 (2016). arXiv: 1611.05431. URL: http://arxiv.org/abs/1611.05431.

[48] Liang Yang et al. "Deep Concrete Inspection Using Unmanned Aerial Vehicle Towards CSSC Database". In: Mar. 2017.

[49] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.